

PATENT

MS147164.1

CERTIFICATE OF MAILING

I hereby certify that this correspondence (along with any paper referred to as being attached or enclosed) is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Date: 2/10/04

Heather Holmes
Heather Holmes

#11
10-28-04
R-J.H./lad

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:

Applicant(s): Nagender P. Vedula, *et al.*

Examiner: Blaine T. Basom

Serial No: 09/662,399

Art Unit: 2173

Filing Date: September 14, 2000

Title: FUNCTION OBJECTS

**Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450**

APPEAL BRIEF

Dear Sir:

Applicants submit this Appeal Brief in triplicate in connection with an appeal of the above-identified patent application. Please charge \$330.00 for the fee associated with this brief to Deposit Account No. 50-1063 [MSFTP128US].

I. Real Party in Interest (37 C.F.R. §1.192(c)(1))

The real party in interest in the present appeal is Microsoft Corporation, the assignee of the present application.

II. Related Appeals and Interferences (37 C.F.R. §1.192(c)(2))

Appellants, appellants' legal representatives, and/or the assignee of the present application are not aware of any appeals or interferences which will directly affect, or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. Status of Claims (37 C.F.R. §1.192(c)(3))

Claims 1-45 are currently pending in the subject application and are presently under consideration. The rejection of claims 1-45 is appealed.

IV. Status of Amendments (37 C.F.R. §1.192(c)(4))

No claim amendments have been entered subsequent the Final Office Action.

V. Summary of Invention (37 C.F.R. §1.192(c)(5))

The subject invention generally relates to mapping a source object with a target object. (See pg. 1, ln. 4). More particularly, the invention relates to function objects, which may be used for defining a mapping between a source object and a target object. (See pg. 1, ln. 5-6, pg. 4, ln. 2-10, pg. 11 ln. 31 – p. 12 ln. 1-5). Function objects allow unsophisticated users to graphically define and compile a mapping between disparate objects without having to create functional computer-executable scripts, thus providing significant advantages over conventional methods (e.g., an ordinary user need not have extensive knowledge in computer programming to generate a mapping between two objects). (See pg. 12 ln. 3-5). The invention further comprises methodologies and tools for creating such a mapping using one or more function objects. (See p. 13, ln. 12-25). In addition, the invention provides a method and tool for creating a function object. (See pg. 20, ln. 6-16 and pg. 22, ln. 3-14). Function objects can include a script component having computer-executable instructions for performing a function (See pg. 12, ln. 21-22). A function object also comprises a graphical component having an input and an output, wherein a user may associate the input with a node related to a first object and the output with a

node related to a second object in creating a graphical mapping between the first object and the second object (*See* pg. 12, ln. 25-28).

VI. Statement of the Issues (37 C.F.R. §1.192(c)(6))

Whether claims 16, 23, 30, 32-35, 37, 38, 41, 42, 43, and 45 are unpatentable under 35 U.S.C. §102(b) as being anticipated by Oppenheim (U.S. 5,734,905).

Whether claims 1-15, 17-22, 24-27, and 44 are unpatentable under 35 U.S.C. §103(a) as being obvious over Oppenheim in view of Microsoft's "Component Model Object Specification."

Whether claims 28, 29, and 31 are unpatentable under 35 U.S.C. §103(a) as being obvious over Oppenheim in view of Jordan (U.S. 5,778,227).

Whether claims 36, 29, and 40 are unpatentable under 35 U.S.C. §103(a) as being obvious over Oppenheim in view of Faustini (U.S. 6,496,870).

VII. Grouping of Claims (37 C.F.R. §1.192(c)(7))

For the purposes of this appeal only, the claims are grouped as follows:

Claims 1-45 stand or fall together.

VIII. Argument (37 C.F.R. §1.192(c)(8))

Rejection of Claims 16, 23, 30, 32-35, 37, 38, 41, 42, 43, and 45 Under 35 U.S.C.

§102(b)

Claims 16, 23, 30, 32-35, 37, 38, 41, 42, 43, and 45 stand rejected under 35 U.S.C. §102(b) as being anticipated by Oppenheim (U.S. 5,734,905). It is submitted that this rejection should be withdrawn for at least the following reasons. Oppenheim does not teach or suggest each and every element of the claimed invention.

A single prior art reference anticipates a patent claim only if it expressly or inherently describes each and every limitation set forth in the patent claim. *Trintec Industries, Inc., v. Top-U.S.A. Corp.*, 295 F.3d 1292, 63 U.S.P.Q.2D 1597 (Fed. Cir. 2002). "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of*

California, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the ... claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).

Regarding independent claims 16, 32, and 42, Oppenheim does not disclose a *function object* utilized to ...*create a mapping... between a source object and a target object* by associating *a source object node* with *a target object node* via the *function object* as recited in these claims and described in the specification See Fig. 2, pg. 13, ln. 7-17. The present invention facilitates constructing a mapping between two object types associated with disparate schemas without requiring an ordinary user to have extensive knowledge in computer programming. Thus, data organized, packaged, and formatted in accordance with one particular schema used by an object (*e.g.*, a schema employed by a particular business or group) can be mapped to an object utilizing a different schema (*e.g.*, a schema utilized by a disparate business or group). The present invention, therefore, facilitates transfer of data across business boundaries without requiring businesses to utilize identical schemas in organizing and packaging such data (*e.g.*, a *mapping* between documents/objects can be created).

Conversely, rather than employing a *function object* to associate *a source object node* with *a target object node*, Oppenheim teaches enabling data to flow through a plurality of disparate *application programs*, wherein a mapping has been previously created to allow flow of such data. More particularly, as provided in Oppenheim: "The present invention also provides a mechanism for linking two or more unrelated *application programs* together in an ongoing manner, thereby allowing the output of one *application program* to automatically flow into another *application program* as its input." (Oppenheim, col. 1, ln. 22-26) (emphasis added). For example, Fig. 8 of Oppenheim illustrates an A/D Converter outputting data to a Signal Processor, and the Signal Processor in turn outputs data to a Filter. The Final Office Action dated September 9, 2003 states that the Signal Processor is a *function object* because "it maps data between... the A/D Converter... and the Filter..." Equating these *application programs* to the claimed *function object*, however, is inconsistent with the definition of *function object* given in the specification.

It is black letter law that a patentee can choose to be his or her own lexicographer by clearly setting forth an explicit definition for a claim term that could differ in scope from that which would be afforded by its ordinary meaning. *The specification acts as a dictionary when it expressly defines terms used in the claims* or when it defines terms by implication. Where the patentee has clearly defined a claim term, that definition usually is dispositive; it is the single best guide to the meaning of a disputed term. *Guttmann, Inc. v. Kopykake Enters.*, 302 F.3d 1352 (Fed. Cir. 2002) (citations omitted) (emphasis added)

In particular, the specification defines a *function object* as an “elemental unit of functional transformation.” See pg. 4, ln. 3-5. Oppenheim, however, discloses that linking *application programs* is utilized when particular *application programs* perform specific operations more desirably than others.

One *application program* might be best for changing the key in which the score or a portion of the score is written, while a second *application program* might be best for modifying the rhythm of the score or for adding special sound effects, and a third *application* might be best for filtering the score (or a corresponding sound file) with a specified function in either the time or frequency domains.

(Oppenheim, col. 1, ln. 46-52) (emphasis added). As the *application programs* disclosed in Oppenheim are designed for a particular task and can act standing alone, they cannot be “an elemental unit of functional transformation”, the definition of the claimed *function object* as defined in the specification.

Further regarding claims 16, 32, and 42, as well as claim 44, Oppenheim does not disclose, teach, or suggest *associating a source object node with an input of a function object* and *associating a target object node with the output of the function object* as recited in the subject claims. Rather, Oppenheim discloses linking three different application programs to allow data to flow from a first application program to a second application program, and from the second application program to a third application program. More particularly, Oppenheim teaches that a first application program outputs data that is received as input data by a second application program. The second application program acts on the received data and outputs the data, wherein the data is employed as input to a third application program. Such linking of *application programs via inputs and outputs* does not anticipate *associating a source object*

node with an input of a function object and associating a target object node with an output of the function object as recited in these claims. *Nodes* are illustrated in connection with the subject invention as structural components of particular schemas employed in objects. See Fig. 2. As Oppenheim teaches the transformation of data *via* disparate *application programs* that do not include *nodes* as recited in these claims, Oppenheim cannot anticipate such claims.

Regarding independent claims 33 and 41, as Oppenheim does not disclose a *function object* employed to create a mapping between a *source object* and a *target object*, Oppenheim cannot disclose a system and method for creating a *function object*. Further regarding claims 33 and 41, as well as independent claims 43 and 45, Oppenheim does not disclose *creating a script component having computer-executable instructions for performing a function using the user interface* in connection with *creating a graphical component associated with the function and having an input and an output* as recited in these claims. Oppenheim discloses enabling “creating an object containing an appropriate mathematical function” in connection with generating a transformer object. See col. 7, lines 1-5. However, the transformer object of Oppenheim does not have an input and an output. Rather, the transformer object is “slapped” onto a transformee object, which creates an entirely new object. However, such transformer object does not include *an input and output*. Oppenheim later discloses *application programs* that have an input and an output, but *creation* of such *application programs* is not disclosed (e.g., the application programs are assumed to be pre-existing). Furthermore, Oppenheim does not teach or suggest *associating the script component, the graphical component, and the interface component*, wherein *the interface component* is *adapted to provide the script component to a compiler in the mapping tool and to provide the graphical component to the graphical user interface*. Such language in these claims refers to creation of a *function object*, and as Oppenheim does not disclose such *function object*, Oppenheim cannot disclose creating a *function object*.

In view of at least the above, it is readily apparent that Oppenheim does not anticipate or make obvious independent claims 16, 32, 33, 41, 42, 43, and 45, and claims 23, 30, 34, 35, 37, and 38, which depend therefrom.

Rejection of Claims 1-15, 17-22, 24-27, and 44 under 35 U.S.C. §103(a)

Claims 1-15, 17-22, 24-27, and 44 are rejected under 35 U.S.C. §103(a) as being unpatentable over Oppenheim and Microsoft's "Component Object Model Specification" (COM Specification). It is respectfully submitted that this rejection should be withdrawn for at least the following reasons. Neither Oppenheim nor Microsoft's COM specification alone or in combination teach or suggest all the claim limitations of the subject invention.

To reject claims in an application under §103, an examiner must establish a *prima facie* case of obviousness. A *prima facie* case of obviousness is established by a showing of three basic criteria. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. See MPEP §706.02(j). The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on applicant's disclosure. See *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991).

Regarding independent claims 1, 16 and 44, as discussed *supra*, Oppenheim does not teach or suggest a *function object* utilized to facilitate creation of a mapping between two objects, let alone particular components of the *function object*. Oppenheim further does not disclose associating a *source object node* with an input of the *function object* and associating a *target object node* with an output of the function object as recited in these claims. Rather, Microsoft's COM specification discusses benefits of object-oriented programming. Thus, like Oppenheim, Microsoft's COM specification does not teach or suggest *mapping a source object* with a *target object via a function object*. Accordingly, this rejection should be withdrawn.

Rejection of Claims 28, 29, and 31 under 35 U.S.C. §103(a)

Claims 28, 29, and 31 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Oppenheim in view of Jordan (US 5,778,227). Reconsideration and allowance of these claims is respectfully requested for at least the following reasons. Jordan fails to make up for the deficiencies of Oppenheim *vis a vis* applicants' claimed invention regarding independent claim

16. Jordan teaches a system that allows application objects to perform operations such as creation, deletion and accessing of database objects, effectively providing an existing application with additional functionality. Jordan, however, does not teach or suggest a *creating a mapping between a source object... and a target object* by employing a *function object* (e.g., an elemental unit of functional transformation).

As claim 16 is believed to be in condition for allowance, the rejection of the subject dependent claims is moot. It is respectfully requested that this rejection be withdrawn.

Rejection of Claims 36, 39, and 40 under 35 U.S.C. §103(a)

Claims 36, 39, and 40 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Oppenheim, and also over Faustini (US 6,496,870). It is respectfully submitted that this rejection should be withdrawn for at least the following reasons.

Faustini, like Oppenheim, does not teach or suggest *creating a function object* that facilitates generating a mapping between two disparate objects associated with different schemas. Rather, Faustini discloses a visual programming environment for object-oriented programming. Thus, Faustine does not make up for the aforementioned deficiencies of Oppenheim regarding independent claim 33, rendering the subject rejection moot. Withdraw of the rejection is therefore respectfully requested.

IX. Conclusion

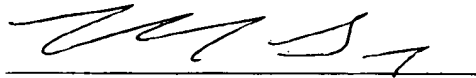
The present application is believed to be in condition for allowance, in view of the above comments. A prompt action to such end is earnestly solicited.

In the event any fees are due in connection with this document, the Commissioner is authorized to charge those fees to Deposit Account No. 50-1063.

Should the Examiner believe a telephone interview would be helpful to expedite favorable prosecution, the Examiner is invited to contact applicant's undersigned representative at the telephone number listed below.

Respectfully submitted,

AMIN & TUROCY, LLP



Himanshu S. Amin

Reg. No. 40,894

AMIN & TUROCY, LLP
24TH Floor, National City Center
1900 E. 9TH Street
Cleveland, Ohio 44114
Telephone (216) 696-8730
Facsimile (216) 696-8731

X. Appendix of Claims (37 C.F.R. § 1.192(c)(9))

1. (Original) A function object for use in creating a mapping in a mapping tool with a graphical user interface, between a source object having a source object node and a target object having a target object node, comprising:

a script component having computer-executable instructions for performing a function;
a graphical component associated with the function, having an input and an output, and adapted to allow a user to graphically associate the input with a source object node and to associate the output with a target object node in the graphical user interface; and

an interface component having a globally unique identifier and adapted to provide the script component to a compiler in the mapping tool and to provide the graphical component to the graphical user interface.

2. (Original) The function object of claim 1, wherein the interface component further includes a category identifier component.

3. (Original) The function object of claim 1, wherein the interface component further comprises a class identifier component.

4. (Original) The function object of claim 1, wherein the interface component further comprises a function category identifier component associated with the function.

5. (Previously presented) The function object of claim 4, wherein the function category identifier component includes one of string, mathematical, logical, date, conversion, scientific, advanced, and custom.

6. (Original) The function object of claim 1, further comprising a second interface component adapted to allow a user to drag and drop the graphical component in the graphical user interface.

7. (Previously presented) The function object of claim 6, wherein the second interface component is associated with the function object by the mapping tool.
8. (Original) The function object of claim 1, further comprising:
 - a plurality of script components having computer-executable instructions for performing a plurality of functions; and
 - a plurality of graphical components individually associated with one of the plurality of functions, individually having an input and an output, and individually adapted to allow a user to graphically associate the input with a source object node and to associate the output with a target object node in the graphical user interface;wherein the interface component is further adapted to provide the plurality of script components to a compiler in the mapping tool and to provide the plurality of graphical components to the graphical user interface.
9. (Previously presented) The function object of claim 8, wherein the interface component further comprises a plurality of function category identifier components individually associated with the plurality of functions.
10. (Previously presented) The function object of claim 9, wherein at least one of the plurality of function category identifier components includes one of string, mathematical, logical, date, conversion, scientific, advanced, and custom.
11. (Original) The function object of claim 10, wherein the plurality of script components having computer-executable instructions for performing a plurality of functions are in a scripting language.
12. (Original) The function object of claim 11, wherein the scripting language is one of java script, visual basic script, and visual C++.
13. (Original) The function object of claim 1, wherein the script component having computer-executable instructions for performing the function is in a scripting language.

14. (Original) The function object of claim 13, wherein the scripting language is one of java script, visual basic script, and visual C++.

15. (Original) The function object of claim 1, wherein the function object is a COM object, and the interface component is a COM interface.

16. (Original) In a mapping tool with a graphical user interface, a method of creating a mapping between a source object having a source object node and a target object having a target object node, the method comprising:

providing a function object having a script component with computer-executable instructions for performing a function, a graphical component associated with the function and having an input and an output, and an interface component;

displaying the graphical component in the user interface;

graphically associating a source object node with the input using a user interface selection device;

graphically associating a target object node with the output using the user interface selection device; and

creating a mapping including the computer-executable instructions, and operative to perform the function according to the source object node, and to provide an output value associated with the target object node according to the function.

17. (Original) The method of claim 16, wherein providing the function object further comprises obtaining the function object from a host computer.

18. (Original) The method of claim 17, wherein obtaining the function object from a host computer further comprises obtaining the function object from a DLL file in the host computer.

19. (Original) The method of claim 18, wherein obtaining the function object from a DLL file in the host computer further comprises:

searching the DLL files in the host computer;
identifying the function object in the searched DLL files; and
loading the function object into the mapping tool.

20. (Original) The method of claim 16, wherein providing the function object further comprises obtaining the function object from a global communications network.

21. (Original) The method of claim 20, wherein obtaining the function object from a global communications network further comprises obtaining the function object from an Internet web site.

22. (Original) The method of claim 21, wherein obtaining the function object from an Internet web site further comprises one of searching the Internet for the function object according to a mapping tool startup script, searching the Internet for the function object according to a user command, and searching a web site provided by the user for the function object.

23. (Original) The method of claim 16, further comprising:
providing a plurality of script components with computer-executable instructions for performing one of a plurality of functions, a plurality of graphical components individually associated with one of the plurality of functions and individually having an input and an output, wherein the interface component is associated with the plurality of script components and the plurality of graphical components; and
displaying at least one of the plurality of graphical components in the user interface.

24. (Original) The method of claim 16, wherein providing the function object further comprises creating a wrapper object in the mapping tool, including the interface component of the function object and a second interface component adapted to allow a user to drag and drop the graphical component in the graphical user interface.

25. (Original) The method of claim 16, wherein displaying the graphical component further comprises obtaining the graphical component from a function object source via the interface component.

26. (Original) The method of claim 25, wherein the function object source is a DLL file on a host computer.

27. (Original) The method of claim 25, wherein obtaining the graphical component from a function object source comprises obtaining the graphical component from a DLL file via the Internet.

28. (Original) The method of claim 16, wherein displaying the graphical component further comprises:

- displaying a function object palette in the graphical user interface;
- displaying the graphical component on the function object palette; and
- allowing a user to drag and drop the graphical component from the function object palette to a mapping screen region in the graphical user interface.

29. (Original) The method of claim 28, further comprising:
providing a plurality of script components with computer-executable instructions for performing one of a plurality of functions, a plurality of graphical components individually associated with one of the plurality of functions and individually having an input and an output, wherein the interface component is associated with the plurality of script components and the plurality of graphical components;

- displaying a function object palette in the graphical user interface;
- displaying at least one of the plurality of graphical components on the function object palette; and
- allowing a user to drag and drop the at least one of the plurality of graphical components from the function object palette to a mapping screen region in the graphical user interface.

30. (Original) The method of claim 16, wherein the mapping tool further comprises a compiler component, and wherein creating the mapping further comprises:

invoking the compiler component to generate compiled output code; and

providing the computer-executable instructions from the script component to the compiler via the interface component.

31. (Original) The method of claim 16, wherein the mapping tool further comprises a compiler component, and wherein creating the mapping further comprises:

providing a plurality of script components with computer-executable instructions for performing one of a plurality of functions, a plurality of graphical components individually associated with one of the plurality of functions and individually having an input and an output, wherein the interface component is associated with the plurality of script components and the plurality of graphical components;

displaying a function object palette in the graphical user interface;

displaying at least one of the plurality of graphical components on the function object palette;

allowing a user to drag and drop the at least one of the plurality of graphical components from the function object palette to a mapping screen region in the graphical user interface;

invoking the compiler component to generate compiled output code; and

providing the computer-executable instructions from the plurality of script components to the compiler via the interface component.

32. (Original) A mapping tool with a graphical user interface for creating a mapping between a source object having a source object node and a target object having a target object node, comprising:

means for providing a function object having a script component with computer-executable instructions for performing a function, a graphical component associated with the function and having an input and an output, and an interface component;

a display device in the graphical user interface adapted to display the graphical component in the user interface;

a user interface selection device adapted to graphically associate a source object node with the input, and to graphically associate a target object node with the output; and

a compiler adapted to create a mapping including the computer-executable instructions, and operative to perform the function according to the source object node, and to provide an output value associated with the target object node according to the function.

33. (Original) In a mapping tool with a graphical user interface, a method of creating a function object for use in creating a mapping between a source object and a target object, the method comprising:

creating a script component having computer-executable instructions for performing a function using the user interface;

creating a graphical component associated with the function and having an input and an output;

creating an interface component adapted to provide the script component to a compiler in the mapping tool and to provide the graphical component to the graphical user interface; and

associating the script component, the graphical component, and the interface component.

34. (Original) The method of claim 33, wherein creating the script component further comprises receiving a user-defined text file including the computer-executable instructions, and creating the script component using the computer-executable instructions from the text file.

35. (Original) The method of claim 34, wherein the computer-executable instructions include one of basic, visual basic, VB script, C++, visual C++, java, java script, and Perl .

36. (Original) The method of claim 33, wherein creating the script component comprises receiving function information from a user, and creating the computer-executable instructions based on the function information.

37. (Original) The method of claim 33, wherein creating the interface component comprises:

receiving a text file from a user, wherein the text file includes information related to the function; and

creating the interface component according to the information in the text file.

38. (Original) The method of claim 37, wherein the information related to the function is in XML.

39. (Original) The method of claim 33, further comprising:
prompting a user for information related to the function object;
receiving prompted information from the user via the graphical user interface;
creating the script component having computer-executable instructions for performing a function using the prompted information; and
creating the graphical component associated with the function and having an input and an output using the prompted information.

40. (Original) The method of claim 39, wherein prompting a user for information related to the function object comprises providing a wizard in the graphical user interface.

41. (Original) A function object creation tool for creating a function object for use in creating a mapping in a mapping tool between a source object and a target object, the function object creation tool comprising:

means for creating a script component having computer-executable instructions for performing a function using the user interface;

means for creating a graphical component associated with the function and having an input and an output;

means for creating an interface component adapted to provide the script component to a compiler in the mapping tool and to provide the graphical component to the graphical user interface; and

means for associating the script component, the graphical component, and the interface component.

42. (Original) A computer-readable medium having computer-executable instructions for performing the following steps:

providing a function object having a script component with computer-executable instructions for performing a function, a graphical component associated with the function and having an input and an output, and an interface component;

displaying the graphical component in a user interface;

graphically associating a source object node with the input using a user interface selection device;

graphically associating a target object node with the output using the user interface selection device; and

creating a mapping including the computer-executable instructions, and operative to perform the function according to the source object node, and to provide an output value associated with the target object node according to the function.

43. (Original) A computer-readable medium having computer-executable instructions for performing the following steps:

creating a script component having computer-executable instructions for performing a function;

creating a graphical component associated with the function and having an input and an output;

creating an interface component adapted to provide the script component to a compiler in a mapping tool and to provide the graphical component to a graphical user interface; and

associating the script component, the graphical component, and the interface component.

44. (Original) A function object for use in creating a mapping, between a source object having a source object node and a target object having a target object node, comprising:

a script component having computer-executable instructions for performing a function;

a graphical component associated with the function, having an input and an output, and adapted to allow a user to graphically associate the input with a source object node and to associate the output with a target object node; and

an interface component having a globally unique identifier and adapted to provide the script component to a compiler and to provide the graphical component to a graphical user interface.

45. (Original) A computer-readable medium having computer-executable instructions for performing the following steps:

creating a script component having computer-executable instructions for performing a function;

creating a graphical component associated with the function and having an input and an output;

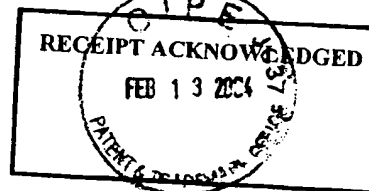
creating an interface component adapted to provide the script component to a compiler and to provide the graphical component to a graphical user interface; and

associating the script component, the graphical component, and the interface component.

Applicant(s) Nagender P. Vedula, et al. Docket No. MS147164.1/MSFTP128US
Serial No. 09/662,399 Filing Date 9/14/00 Atty(s) HSA/MM
Title FUNCTION OBJECTS
Mailed 2/10/2004 Due Date _____ Exp. Mail No. _____

- ☐ Patent Application (☐ New ☐ CON ☐ DIV ☐ CIP ☐ CPA ☐ PCT ☐ Provisional)
☐ Transmittal/Request ☐ Declaration/POA ☐ Issue Fee Transmittal
— Pages of Specification ☐ Assign. & Cover Sheet ☐ Main. Fee Transmittal
— Sheets of Drawings ☐ Small Entity Statement ☐ Ltr. Off. Draftsman
☐ Formal ☐ Informal ☐ IDS/1449/Refs. ☐ Notice of Appeal
— Pages of Abstract ☐ Priority Documents ☒ Appeal Brief
☐ Express Mail Certificate ☐ Check \$ _____ \$ _____
☐ Amendment/Response ☐ Transmittal ☐ Ext. of Time
☐ Trademark/Service App. ☐ Specimens ☐ St. of Use

Other: _____



DOCKETED
FEB 25 2004

RAM Fee History Query Revenue Accounting and Management

Name/Number: 09662399

Total Records Found: 7

Start Date: Any Date

End Date: Any Date

Accounting Date	Sequence Num.	Tran Type	Fee Code	Fee Amount	Mailroom Date	Payment Method
09/28/2000	00000004	<u>1</u>	<u>101</u>	\$690.00	09/14/2000	CK
09/28/2000	00000005	<u>1</u>	<u>103</u>	\$450.00	09/14/2000	CK
09/28/2000	00000006	<u>1</u>	<u>102</u>	\$468.00	09/14/2000	CK
09/28/2000	00000007	<u>1</u>	<u>581</u>	\$40.00	09/14/2000	CK
06/27/2003	00000001	<u>1</u>	<u>1251</u>	\$110.00	06/24/2003	DA 501063
12/19/2003	00000002	<u>1</u>	<u>1401</u>	\$330.00	12/10/2003	DA 501063
02/18/2004	00000131	<u>1</u>	<u>1402</u>	\$330.00	02/13/2004	DA 501063